File created: 12-Jun-90 17:46:11 **{DSK}<usr>local>lde>lispcore>library>TCPUDP.;2**

changes to: (VARS TCPUDPCOMS)

previous date: 6-Jan-89 16:37:55 {DSK}<usr>local>lde>lispcore>library>TCPUDP.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ TCPUDPCOMS
       [(COMS ;; User Datagram Protocol --- Definitions
              [DECLARE%: EVAL@COMPILE DONTCOPY (EXPORT (RECORDS UDP)
                                                       (CONSTANTS (\UDPOVLEN 8]
              (FILES (SYSLOAD)
                     TCPLLIP))
        (COMS ;; Internal functions
              (FNS UDP.GET.BYTE UDP.GET.CELL UDP.GET.STRING UDP.GET.WORD \UDP.FLUSH.SOCKET.QUEUE
                   \UDP.PORTCOMPARE \UDP.CHECKSUM \UDP.SET.CHECKSUM)
              (FNS \UDP.HANDLE.ICMP))
        (COMS ;; External functions
              (FNS PRINTUDP UDP.INIT UDP.STOP UDP.OPEN.SOCKET UDP.CLOSE.SOCKET UDP.SOCKET.EVENT
                   UDP.SOCKET.NUMBER UDP.GET UDP.SEND UDP.EXCHANGE UDP.SETUP UDP.APPEND.BYTE UDP.APPEND.CELL
                   UDP.APPEND.STRING UDP.APPEND.WORD UDP.INCREMENT.LENGTH)
              (ADDVARS (IPPRINTMACROS (17 . PRINTUDP)))
              (DECLARE%: DONTEVAL@LOAD DOCOPY (P (MOVD? 'NILL 'PRINTRPCDATA)
                                                 (UDP.INIT])
```

;; User Datagram Protocol --- Definitions

```
(DECLARE%: EVAL@COMPILE DONTCOPY
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

[ACCESSFNS UDP ((UDPBASE (\IPDATABASE DATUM)))
       (BLOCKRECORD UDPBASE ((UDPSOURCEPORT WORD)
                             (UDPDESTPORT WORD)
                             (UDPLENGTH WORD)
                             (UDPCHECKSUM WORD)))
       (ACCESSFNS UDP ((UDPCONTENTS (\ADDBASE (\IPDATABASE DATUM)
                                              (FOLDHI \UDPOVLEN BYTESPERWORD]
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \UDPOVLEN 8)

(CONSTANTS (\UDPOVLEN 8))
)
)
```

;; END EXPORTED DEFINITIONS

```
(FILESLOAD (SYSLOAD)
       TCPLLIP)
```

;; Internal functions

```
(DEFINEQ
```

(UDP.GET.BYTE

```
  [LAMBDA (UDP BYTE#)                                    (* ejs%: "25-Jun-85 21:04")
```

(* * Return a byte from the UDP data area)

```
    (COND
       ((AND (IGEQ BYTE# 0)
             (ILESSP BYTE# (fetch (UDP UDPLENGTH) of UDP)))
        (\GETBASEBYTE (fetch (UDP UDPCONTENTS) of UDP)
               BYTE#])
```

(UDP.GET.CELL

```
  [LAMBDA (UDP CELL#)                                    (* ejs%: "25-Jun-85 21:09")
```

```
        (* * Return a cell from the UDP data area)

    (COND
       ((AND (IGEQ CELL# 0)
             (ILESSP CELL# (FOLDLO (fetch (UDP UDPLENGTH) of UDP)
                                   BYTESPERCELL)))
        (\MAKENUMBER (\GETBASE (fetch (UDP UDPCONTENTS) of UDP)
                               (UNFOLD CELL# WORDSPERCELL))
               (\GETBASE (fetch (UDP UDPCONTENTS) of UDP)
                      (ADD1 (UNFOLD CELL# WORDSPERCELL])
```

(UDP.GET.STRING

```
  [LAMBDA (UDP OFFSET)                                    (* ejs%: "25-Jun-85 21:12")

        (* * Fetch a string out of the UDP packet)

    (OR (SMALLP OFFSET)
        (SETQ OFFSET 0))
    (LET* ((LENGTH (IDIFFERENCE (fetch (UDP UDPLENGTH) of UDP)
                                OFFSET))
           (STRING (ALLOCSTRING LENGTH)))
          (\MOVEBYTES (fetch (UDP UDPCONTENTS) of UDP)
                 OFFSET
                 (fetch (STRINGP BASE) of STRING)
                 (fetch (STRINGP OFFST) of STRING)
                 LENGTH)
          STRING])
```

(UDP.GET.WORD

```
  [LAMBDA (UDP WORD#)                                     (* ejs%: "25-Jun-85 21:06")

        (* * Return a word from the UDP data area)

    (COND
       ((AND (IGEQ WORD# 0)
             (ILESSP WORD# (FOLDLO (fetch (UDP UDPLENGTH) of UDP)
                                   BYTESPERWORD)))
        (\GETBASE (fetch (UDP UDPCONTENTS) of UDP)
               WORD#])
```

(\UDP.FLUSH.SOCKET.QUEUE

```
  [LAMBDA (IPSOCKET)                                      ; Edited 25-Aug-88 12:57 by bvm

;;; Called to flush input packet queue on an IPSOCKET

    (LET ((QUEUE (fetch (IPSOCKET IPSQUEUE) of IPSOCKET))
          PACKET)
         (UNINTERRUPTABLY
             (while (SETQ PACKET (\DEQUEUE QUEUE)) do (\RELEASE.ETHERPACKET PACKET)
                finally (replace (IPSOCKET IPSQUEUELENGTH) of IPSOCKET with 0)))])
```

(\UDP.PORTCOMPARE

```
  [LAMBDA (UDP IPSOCKET)                                  (* ejs%: " 9-Feb-85 14:37")

        (* * Compare IPSOCKET until we find the one this UDP was destined for)

    (EQ (fetch (UDP UDPDESTPORT) of UDP)
        (fetch (IPSOCKET IPSOCKET) of IPSOCKET])
```

(\UDP.CHECKSUM

```
  [LAMBDA (UDP ZeroCheckSumIsOK)                          (* HAS%: "19-Aug-86 16:47")

        (* * Compute the UDP checksum for the packet UDP. The packet is assumed to have been setup by UDP.SETUP so that
        source and destination addresses, protocol, and UDP length have already been set.)

    (COND
       ((AND ZeroCheckSumIsOK (EQ (fetch (UDP UDPCHECKSUM) of UDP)
                                  0))

        (* * BSD Unix strikes again!)

        0)
       (T (LET ((SOURCE (fetch (IP IPSOURCEADDRESS) of UDP))
                (DEST (fetch (IP IPDESTINATIONADDRESS) of UDP))
                (LENGTH (fetch (UDP UDPLENGTH) of UDP))
                CHECKSUM)
               (SETQ CHECKSUM (IPLUS (bind (BASE _ (LOCF (fetch (IP IPSOURCEADDRESS) of UDP))) for I from 0
                                        to (CONSTANT (SUB1 (TIMES 2 WORDSPERCELL)))
                                        sum (\GETBASE BASE I))
                                     (ffetch (IP IPPROTOCOL) of UDP)
                                     LENGTH
```

```
                                    (\IPCHECKSUM UDP (\IPDATABASE UDP)
                                       LENGTH)))
              (SETQ CHECKSUM (IPLUS (LDB (BYTE 16 16)
                                         CHECKSUM)
                                    (LDB (BYTE 16 0)
                                         CHECKSUM)))
              [COND
                 ((NOT (EQ (LDB (BYTE 16 16)
                                CHECKSUM)
                           0))
                  (SETQ CHECKSUM (IPLUS (LDB (BYTE 16 16)
                                             CHECKSUM)
                                        (LDB (BYTE 16 0)
                                             CHECKSUM]
              CHECKSUM])
```

(**UDP.SET.CHECKSUM**

```
  [LAMBDA (UDP)                                                    (* ejs%: " 3-Jun-85 00:19")

          (* * Called to set the UDP checksum in a packet ready to be transmitted)

    (LET (CHECKSUM)
         (replace (UDP UDPCHECKSUM) of UDP with 0)
         (SETQ CHECKSUM (\UDP.CHECKSUM UDP))
         (replace (UDP UDPCHECKSUM) of UDP with (COND
                                                   [(NEQ CHECKSUM MAX.SMALLP)
                                                    (LOGAND (LOGNOT CHECKSUM)
                                                            (CONSTANT (MASK.1'S 0 16]
                                                   (T MAX.SMALLP])

)

(DEFINEQ
```

(**UDP.HANDLE.ICMP**

```
  [LAMBDA (ICMP SENTIP PROTOCOL)                                   ; Edited 13-Sep-88 14:26 by bvm
    ;; Handle an ICMP packet sent to a UDP socket.  We allow each UDP client to decide how to handle these.
    (LET ((SOCKET (\IP.FIND.SOCKET (ffetch (UDP UDPSOURCEPORT) of SENTIP)
                                   PROTOCOL))
          FN)
         (if (OR (NULL SOCKET)
                 (EQ (SETQ FN (ffetch (IPSOCKET IPSICMPFN) of SOCKET))
                     '\UDP.HANDLE.ICMP))
             then                                                  ; Sender went away already, or else didn't specify a handler (so
                                                                   ; inherited the default)
                  (\RELEASE.ETHERPACKET ICMP)
           else (CL:FUNCALL FN ICMP SENTIP SOCKET])

)

;; External functions

(DEFINEQ
```

(**PRINTUDP**

```
  [LAMBDA (UDP FILE)                                               ; Edited  6-Jan-89 16:18 by Briggs
    (printout FILE "UDP Source port: " (fetch (UDP UDPSOURCEPORT) of UDP)
           " Dest port: "
           (fetch (UDP UDPDESTPORT) of UDP)
           T "Length: " (fetch (UDP UDPLENGTH) of UDP)
           " Checksum: "
           (fetch (UDP UDPCHECKSUM) of UDP)
           T)
    (COND
       ((OR (EQ (fetch (UDP UDPDESTPORT) of UDP)
                \TFTP.SOCKET)
            (EQ (fetch (UDP UDPSOURCEPORT) of UDP)
                \TFTP.SOCKET))
        (PRINTTFTP UDP FILE))
       (T (PRINTRPCDATA (fetch (UDP UDPCONTENTS) of UDP)
                 (- (fetch (UDP UDPLENGTH) of UDP)
                    \UDPOVLEN)
                 FILE])
```

(**UDP.INIT**

```
  [LAMBDA NIL                                                      ; Edited 25-Aug-88 12:54 by bvm
    (COND
       ((OR \IPFLG (SELECTQ (ASKUSER 15 'Y "IP is not running.  Shall I attempt to initialize it? ")
                            (Y (\IPINIT)
                               \IPFLG)
                            NIL))
        (\IP.ADD.PROTOCOL \UDP.PROTOCOL (FUNCTION \UDP.PORTCOMPARE)
               NIL NIL (FUNCTION \UDP.HANDLE.ICMP])
```

(UDP.STOP

```
  [LAMBDA NIL                                                    (* ejs%: " 9-Feb-85 14:43")
    (\IP.DELETE.PROTOCOL \UDP.PROTOCOL])
```

(UDP.OPEN.SOCKET

```
  [LAMBDA (SKT# IFCLASH ICMPFN)                                  ; Edited 25-Aug-88 13:03 by bvm
    (LET ((UDPCHAIN (\IP.FIND.PROTOCOL \UDP.PROTOCOL)))
         (if (OR UDPCHAIN (SETQ UDPCHAIN (UDP.INIT)))
             then [if (NULL SKT#)
                      then                                       ; Open any free socket
                           (\IP.OPEN.SOCKET \UDP.PROTOCOL NIL NIL NIL NIL NIL ICMPFN)
                    else                                         ; Check for clash
                         (LET ((IPSOCKET (\IP.FIND.SOCKET SKT# UDPCHAIN)))
                              (if (NULL IPSOCKET)
                                  then (\IP.OPEN.SOCKET \UDP.PROTOCOL SKT# NIL NIL NIL NIL ICMPFN)
                                else (SELECTQ IFCLASH
                                              ((T ACCEPT)
                                                   (\UDP.FLUSH.SOCKET.QUEUE IPSOCKET)
                                                   IPSOCKET)
                                              ((DON'T FAIL)
                                                   NIL)
                                              (ERROR "UDP Port is already in use" SKT#]
              else                                               ; IP not inited
                   (SELECTQ IFCLASH
                            ((DON'T FAIL)
                                 NIL)
                            (ERROR!])
```

(UDP.CLOSE.SOCKET

```
  [LAMBDA (IPSOCKET NOERRORFLG)                                  (* ejs%: " 9-Feb-85 15:00")
    (\UDP.FLUSH.SOCKET.QUEUE IPSOCKET)
    (\IP.CLOSE.SOCKET (fetch (IPSOCKET IPSOCKET) of IPSOCKET)
           \UDP.PROTOCOL NOERRORFLG])
```

(UDP.SOCKET.EVENT

```
  [LAMBDA (IPSOCKET)                                             (* ejs%: " 9-Feb-85 15:07")
    (fetch (IPSOCKET IPSEVENT) of IPSOCKET])
```

(UDP.SOCKET.NUMBER

```
  [LAMBDA (IPSOCKET)                                             (* ejs%: " 9-Feb-85 15:08")
    (fetch (IPSOCKET IPSOCKET) of IPSOCKET])
```

(UDP.GET

```
  [LAMBDA (IPSOCKET WAIT)                                        ; Edited 13-Sep-88 11:59 by bvm
```

;;; Returns the next UDP packet on the queue, or NIL if none exist and WAIT is NIL. If WAIT is T, this function waits forever. If WAIT is an integer, it is interpreted as the number of milliseconds to wait before returning NIL or a packet which arrives during that time. This function therefore is like GETXIP and GETPUP

```
    (PROG ((QUEUE (fetch (IPSOCKET IPSQUEUE) of IPSOCKET))
           UDP TIMER)
      LP   (UNINTERRUPTABLY
               (COND
                  ((SETQ UDP (\DEQUEUE QUEUE))
                   (add (fetch (IPSOCKET IPSQUEUELENGTH) of IPSOCKET)
                        -1))))
           (COND
              [(NULL UDP)
               (COND
                  (WAIT (COND
                            ((EQ WAIT T)                         ; Wait forever

                             )
                            [TIMER (COND
                                       ((TIMEREXPIRED? TIMER)
                                        (RETURN]
                            (T (OR (FIXP WAIT)
                                   (LISPERROR "NON-NUMERIC ARG" WAIT))
                               (SETQ TIMER (SETUPTIMER WAIT))
                               T))
                        (AWAIT.EVENT (fetch (IPSOCKET IPSEVENT) of IPSOCKET)
                               TIMER T)
                        (GO LP))
                  (T (BLOCK]
              ([AND (EQ (fetch (IP IPPROTOCOL) of UDP)
                        \UDP.PROTOCOL)
                    (NEQ (fetch (UDP UDPCHECKSUM) of UDP)
                         0)
                    (NOT (\IP.CHECKSUM.OK (\UDP.CHECKSUM UDP]    ; Bad checksum on UDP packet. Any other kind of packet must
                                                                 ; have been put there by someone else
               (\RELEASE.ETHERPACKET UDP)
```

```
            (GO LP)))
          (RETURN UDP])
```

(UDP.SEND

```
  [LAMBDA (IPSOCKET UDP)                                    (* ejs%: " 9-Feb-85 15:24")

          (* * Sends a UDP packet. IP and UDP header assumed set up by UDP.SETUP and \IP.SETUPIP)

    (\UDP.SET.CHECKSUM UDP)
    (\IP.TRANSMIT UDP])
```

(UDP.EXCHANGE

```
  [LAMBDA (IPSOCKET OUTUDP TIMEOUT)                         (* ejs%: " 9-Feb-85 22:28")

          (* * Send a UDP packet and wait for TIMEOUT to receive a packet
          (TIMEOUT defaults to \ETHERTIMEOUT))

    (\UDP.FLUSH.SOCKET.QUEUE IPSOCKET)
    (UDP.SEND IPSOCKET OUTUDP)
    (BLOCK)
    (UDP.GET IPSOCKET (OR (FIXP TIMEOUT)
                          \ETHERTIMEOUT])
```

(UDP.SETUP

```
  [LAMBDA (UDP DESTHOST DESTSOCKET ID IPSOCKET REQUEUE)     (* ejs%: " 9-Feb-85 16:04")
    (\IP.SETUPIP UDP DESTHOST ID IPSOCKET REQUEUE)
    (add (fetch (IP IPTOTALLENGTH) of UDP)
         \UDPOVLEN)
    (AND (SMALLP DESTSOCKET)
         (replace (UDP UDPDESTPORT) of UDP with DESTSOCKET))
    (replace (UDP UDPSOURCEPORT) of UDP with (fetch (IPSOCKET IPSOCKET) of IPSOCKET))
    (replace (UDP UDPLENGTH) of UDP with \UDPOVLEN)
   UDP])
```

(UDP.APPEND.BYTE

```
  [LAMBDA (UDP BYTE)                                        (* ejs%: " 9-Feb-85 16:07")
    (\IP.APPEND.BYTE UDP BYTE)
    (add (fetch (UDP UDPLENGTH) of UDP)
         1])
```

(UDP.APPEND.CELL

```
  [LAMBDA (UDP CELL)                                        (* ejs%: " 9-Feb-85 16:06")
    (\IP.APPEND.CELL UDP CELL)
    (add (fetch (UDP UDPLENGTH) of UDP)
         BYTESPERCELL])
```

(UDP.APPEND.STRING

```
  [LAMBDA (UDP STRING)                                      (* ejs%: " 9-Feb-85 16:10")
    (OR (STRINGP STRING)
        (SETQ STRING (MKSTRING STRING)))
    (\IP.APPEND.STRING UDP STRING)
    (add (fetch (UDP UDPLENGTH) of UDP)
         (NCHARS STRING])
```

(UDP.APPEND.WORD

```
  [LAMBDA (UDP WORD)                                        (* ejs%: " 9-Feb-85 16:07")
    (\IP.APPEND.WORD UDP WORD)
    (add (fetch (UDP UDPLENGTH) of UDP)
         WORDSPERCELL])
```

(UDP.INCREMENT.LENGTH

```
  [LAMBDA (UDP INCREMENT)                                   (* ejs%: "12-Apr-86 18:50")
    (add (fetch (IP IPTOTALLENGTH) of UDP)
         INCREMENT)
    (add (fetch (UDP UDPLENGTH) of UDP)
         INCREMENT)
   INCREMENT])

)

(ADDTOVAR IPPRINTMACROS (17 . PRINTUDP))

(DECLARE%: DONTEVAL@LOAD DOCOPY

(MOVD? 'NILL 'PRINTRPCDATA)

(UDP.INIT)
)
```

FUNCTION INDEX

VARIABLE INDEX

CONSTANT INDEX

RECORD INDEX